LAPORAN PRAKTIKUM
MODUL 7
QUEUE

Disusun oleh:

Aji Tri Prasetyo

NIM : 2311102064

Dosen Pengampu:

Wahyu Andi Saputra, S.Pd., M.Eng.

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa mampu menjelaskan definisi dan konsep dari double queue
2. Mahasiswa mampu menerapkan operasi tambah, menghapus pada queue
3. Mahasiswa mampu menerapkan operasi tampil data pada queue

BAB II
DASAR TEORI

Pengertian Queue

Queue adalah struktur data linier yang menerapkan prinsip operasi dimana elemen data yang masuk pertama akan keluar lebih dulu. Prinsip ini dikenal dengan istilah FIFO (First In, First Out).

Berbeda dengan struktur data stack yang menyimpan data secara bertumpuk dimana hanya terdapat satu ujung yang terbuka untuk melakukan operasi data, struktur data queue justru disusun secara horizontal dan terbuka di kedua ujungnya. Ujung pertama (head) digunakan untuk menghapus data sedangkan ujung lainnya (tail) digunakan untuk menyisipkan data.

Persamaan antara stack dan queue adalah keduanya dapat diimplementasikan menggunakan struktur data linked list atau array. Contoh nyata dalam kehidupan sehari-hari yang dapat menggambarkan struktur data queue adalah barisan orang yang menunggu untuk membeli tiket di gedung bioskop.

Orang yang baru datang akan bergabung dengan barisan dari ujung dan orang yang berdiri di depan akan menjadi yang pertama mendapatkan tiket dan meninggalkan barisan. Demikian pula dalam struktur data queue, data yang ditambahkan terlebih dahulu akan meninggalkan antrian terlebih dahulu.

BAB III

GUIDED

1. GUIDED 1

SOURCE CODE

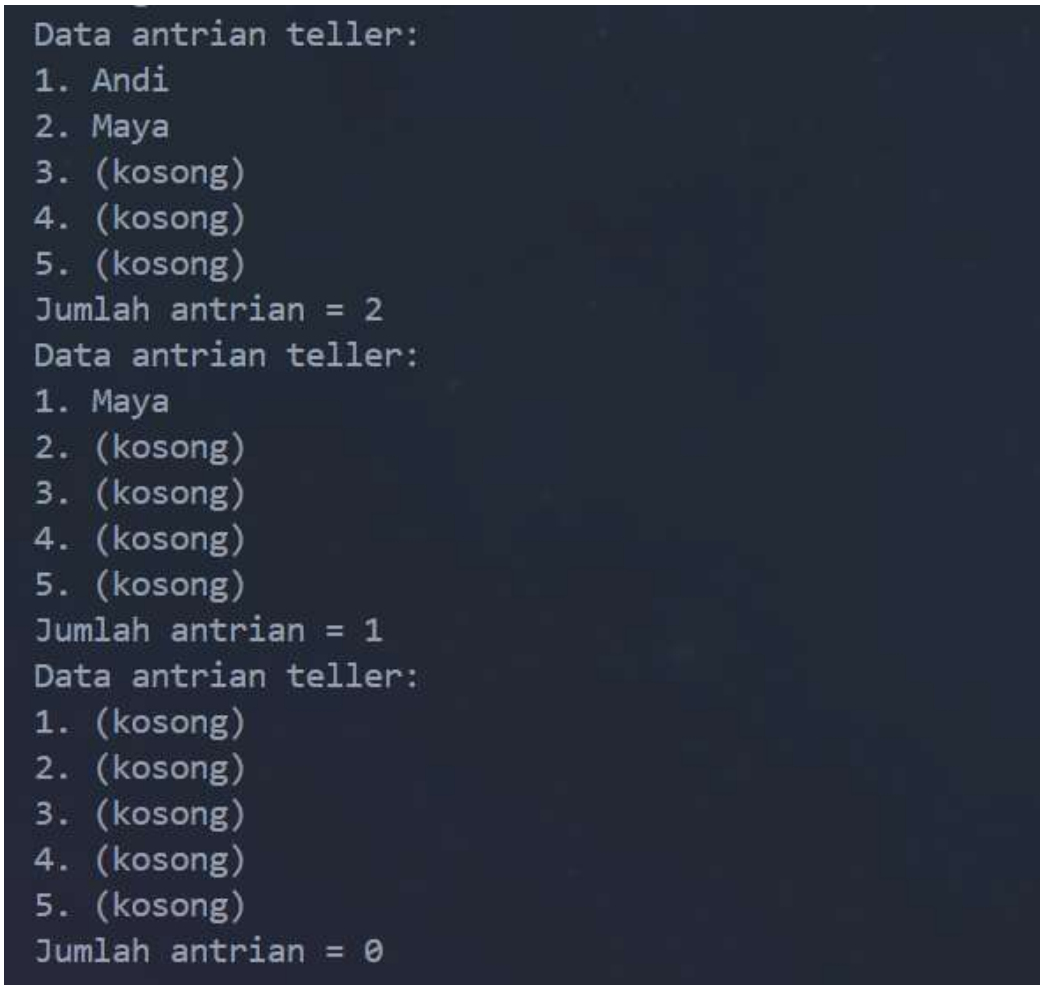
```
#include <iostream>
using namespace std;

const int maksimalQueue = 5; // Maksimal antrian
int front = 0;               // Penanda antrian
int back = 0;                // Penanda
string queueTeller[5];       // Fungsi pengecekan

bool isFull()
{ // Pengecekan antrian penuh atau tidak
    if (back == maksimalQueue)
    {
        return true; // =1
    }
    else
    {
        return false;
    }
}

bool isEmpty()
{ // Antriannya kosong atau tidak
    if (back == 0)
    {
        return true;
    }
    else
    {
```

```
        return false;
    }
}

void enqueueAntrian(string data)
{ // Fungsi menambahkan antrian
    if (isFull())
    {
        cout << "Antrian penuh" << endl;
    }
    else
    {
        if (isEmpty())
        { // Kondisi ketika queue kosong
            queueTeller[0] = data;
            front++;
            back++;
        }
        else
        { // Antrianya ada isi
            queueTeller[back] = data;
            back++;
        }
    }
}

void dequeueAntrian()
{ // Fungsi mengurangi antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
```

```
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = queueTeller[i + 1];
        }
        back--;
    }
}

int countQueue()
{ // Fungsi menghitung banyak antrian
    return back;
}

void clearQueue()
{ // Fungsi menghapus semua antrian
    if (isEmpty())
    {
        cout << "Antrian kosong" << endl;
    }
    else
    {
        for (int i = 0; i < back; i++)
        {
            queueTeller[i] = "";
        }
        back = 0;
        front = 0;
    }
}

void viewQueue()
{ // Fungsi melihat antrian
    cout << "Data antrian teller:" << endl;
    for (int i = 0; i < maksimalQueue; i++)
```

```
    {
        if (queueTeller[i] != "")
        {
            cout << i + 1 << ". " << queueTeller[i] << endl;
        }
        else
        {
            cout << i + 1 << ". (kosong)" << endl;
        }
    }
}

int main()
{
    enqueueAntrian("Andi");
    enqueueAntrian("Maya");
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    dequeueAntrian();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    clearQueue();
    viewQueue();
    cout << "Jumlah antrian = " << countQueue() << endl;
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Kode ini merupakan implementasi dari struktur data Queue menggunakan array di bahasa pemrograman C++. Queue adalah struktur data yang menggunakan metode FIFO (First-In First-Out), yang berarti elemen pertama yang masuk adalah elemen pertama yang keluar.

UNGUIDED

1. UNGUIDED 1

Ubahlah penerapan konsep queue pada bagian guided dari array menjadi linked list

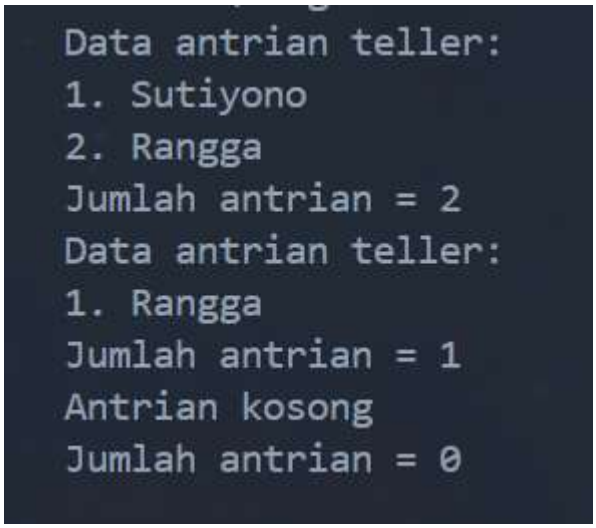
SOURCE CODE

```
#include <iostream>
using namespace std;

struct Node {
    string data;
    Node* next;
};

class Queue {
private:
    Node* front;
    Node* back;

public:
    Queue() {
        front = NULL;
        back = NULL;
    }

    bool isEmpty() {
        return front == NULL;
    }

    void enqueue(string data) {
        Node* newNode = new Node;
        newNode->data = data;
```

```
        newNode->next = NULL;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            Node* temp = front;
            front = front->next;
            delete temp;
        }
    }

    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != NULL) {
            count++;
            current = current->next;
        }
        return count;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeue();
```

```
        }
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            cout << "Data antrian teller:" << endl;
            Node* current = front;
            int position = 1;
            while (current != NULL) {
                cout << position << ". " << current->data <<
endl;
                current = current->next;
                position++;
            }
        }
    }
};

int main() {
    Queue queue;
    queue.enqueue("Sutiyono");
    queue.enqueue("Rangga");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

enqueue: Menambahkan elemen baru ke dalam antrian, Membuat node baru dan menginisialisasi data, Jika antrian kosong, front dan back diatur ke node baru, Jika tidak, node baru ditambahkan ke akhir antrian dan back diperbarui..

2. UNGUIDED 1

Dari nomor 1 buatlah konsep antri dengan atribut Nama mahasiswa dan NIM Mahasiswa

SOURCE CODE

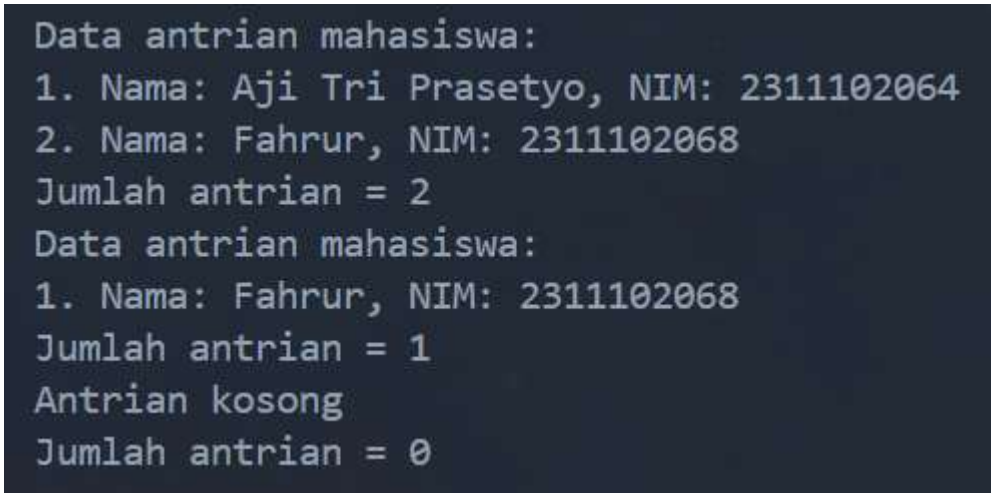
```
#include <iostream>
using namespace std;



struct Node {
    string nama;
    string nim;
    Node* next;
};
```

```
class Queue {
private:
    Node* front;
    Node* back;

public:
    Queue() {
        front = NULL;
        back = NULL;
    }

    bool isEmpty() {
        return front == NULL;
    }

    void enqueue(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = NULL;

        if (isEmpty()) {
            front = newNode;
            back = newNode;
        } else {
            back->next = newNode;
            back = newNode;
        }
    }

    void dequeue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
```

```
        } else {
            Node* temp = front;
            front = front->next;
            delete temp; }
    }

    int countQueue() {
        int count = 0;
        Node* current = front;
        while (current != NULL) {
            count++;
            current = current->next;
        }
        return count;
    }

    void clearQueue() {
        while (!isEmpty()) {
            dequeue();
        }
    }

    void viewQueue() {
        if (isEmpty()) {
            cout << "Antrian kosong" << endl;
        } else {
            cout << "Data antrian mahasiswa:" << endl;
            Node* current = front;
            int position = 1;
            while (current != NULL) {
                cout << position << ". Nama: " << current->nama
<< ", NIM: " << current->nim << endl;
                current = current->next;
                position++;
```

```
            }
        }
    }
};

int main() {
    Queue queue;
    queue.enqueue("Aji Tri Prasetyo", "2311102064");
    queue.enqueue("Fahrur", "2311102068");
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.dequeue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    queue.clearQueue();
    queue.viewQueue();
    cout << "Jumlah antrian = " << queue.countQueue() << endl;
    return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Secara keseluruhan, kode ini menunjukkan implementasi Queue menggunakan linked list untuk menyimpan dan mengelola data mahasiswa dengan operasi dasar seperti menambahkan, menghapus, menghitung, mengosongkan, dan menampilkan elemen dalam antrian. Implementasi ini lebih fleksibel dibandingkan dengan penggunaan array karena tidak ada batasan tetap pada jumlah elemen yang dapat disimpan dalam antrian (selama ada memori yang tersedia).

BAB IV
KESIMPULAN

Setelah melakukan pembelajaran mengenai Queue di Bahasa Pemrograman C++ berikut poin utama yang telah dipelajari :

1. Queue adalah struktur data yang mengikuti prinsip FIFO (First-In First-Out), di mana elemen yang pertama kali dimasukkan akan menjadi elemen pertama yang dikeluarkan.
2. Materi tentang queue di C++ mencakup berbagai aspek dari implementasi dan penggunaan queue, baik menggunakan array maupun linked list.

DAFTAR PUSTAKA

Trivusi. (2023, 01 Juli) Struktur Data Queue: Pengertian, Jenis, dan Kegunaannya. diakses pada 26 Mei 2024 dari https://www.trivusi.web.id/2022/07/struktur-data-queue.html